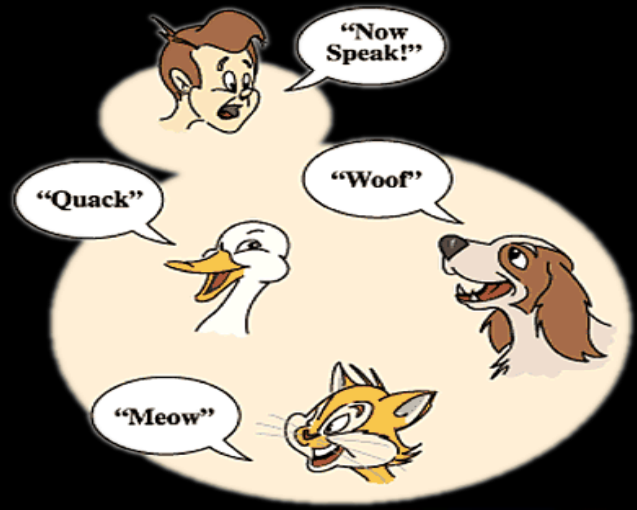
Polymorphism

Abstract Classes and Methods, Overriding



SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Contents

Questions

SoftUni Foundation

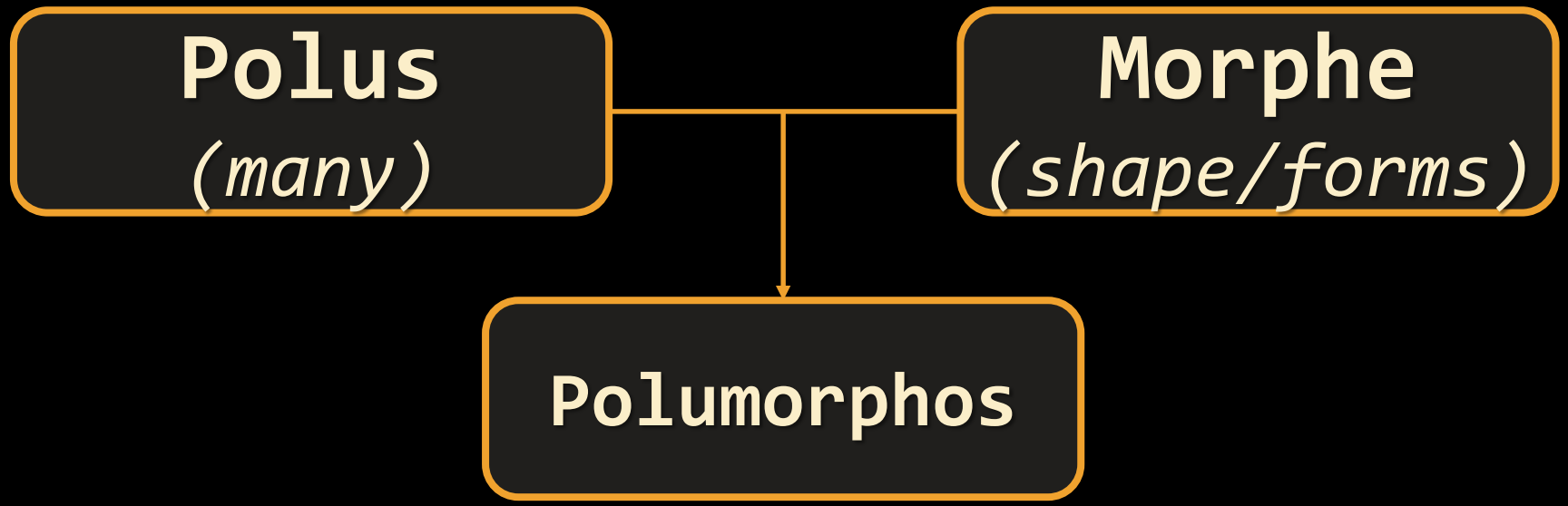
sli.do

#JavaFundamentals

Polymorphism

What is Polimorphism?

- From the Greek



This is something similar to word having **several different** meanings **depending** on the **context**

Polymorphism in OOP

SoftUni Foundation

- Ability of an **object** to take on **many forms**

```
public interface Animal {}
public abstract class Mammal {}
public class Person extends Mammal implements Animal {}
```

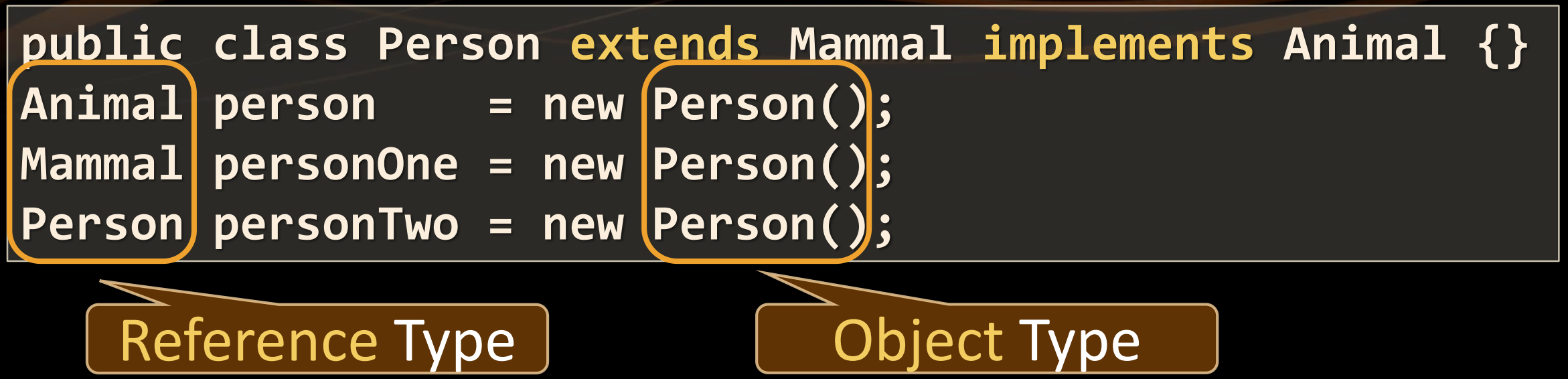
```
Person IS-A Person
```

```
Person IS-AN Animal
```

```
Person IS-A Mammal
```

```
Person IS-AN Object
```

Reference Type and Object Type



- **Variables** are saved in **reference** type
- You can use only **reference methods**
- If you need **object method** you need to **cast it or override it**

Keyword - instanceof

SoftUni Foundation

- Check if **object** is an **instance** of a specific **class**

```
Mammal gosho = new Person();
Person ivan = new Person();
```

```
if (person instanceof Person) {
   ((Person) person).getSalary();
}
```

Check object type of person

```
if (person.getClass() == Person.class) {
   ((Person) person).getSalary();
}
```

Cast to object type and use its methods

Keyword - instanceof (2)

SoftUni Foundation

```
Anytime you find yourself writing code of the form "if
the object is of type T1, then do something, but if
it's of type T2, then do something else"…

                    slap yourself.
```

From *Effective C++*, by Scott Meyers

Types of Polymorphism

SoftUni Foundation

- **Runtime** polymorphism

```
public class Shape {}
public class Circle extends Shape {}
public static void main(String[] args) {
  Shape shape = new Circle()
}
```

Method overriding

- **Compile time** polymorphism

```
public static void main(String[] args) {
  int sum(int a, int b, int c)
  double sum(Double a, Double b)
}
```

Method overloading

Compile Time Polymorphism

SoftUni Foundation

- Also known as **Static Polymorphism**

```
public static void main(String[] args) {
  static int myMethod(int a, int b) {}
  static Double myMethod(Double a, Double b)
}
```

Method overloading

- Argument lists could differ in:
  - Number of parameters
  - Data type of parameters
  - Sequence of Data type of parameters

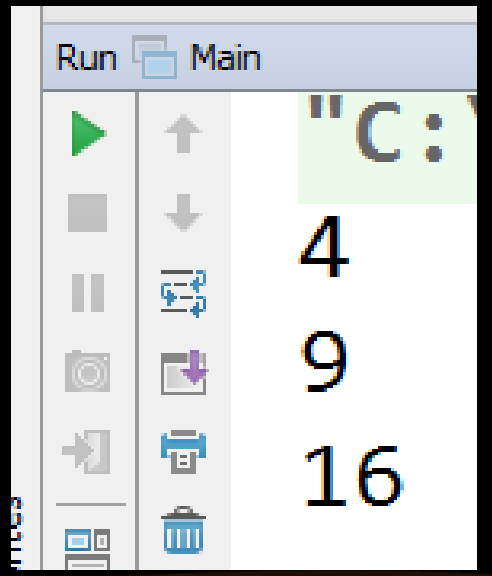
Problem: Overload Method

SoftUni Foundation

MathOperation
+add(int a, int b): int +add(int a, int b, int c): int +add(int a, int b, int c, int d): int

```
MathOperation mathOperation = new MathOperation();

System.out.println(mathOperation.add( a: 2,  b: 2));
System.out.println(mathOperation.add( a: 3,  b: 3,  c: 3));
System.out.println(mathOperation.add( a: 4,  b: 4,  c: 4,  d: 4));
```



Solution: Overload Method

SoftUni Foundation

```
public class MathOperation {
  public int add(int a, int b) {
    return a + b;
  }
  public int add(int a, int b, int c) {
    return a + b + c;
  }
  public int add(int a, int b, int c, int d) {
    return a + b + c + d;
  }
}
```

Rules for Overloading Method

SoftUni Foundation

- **Overloading** can take place in the **same class** or in its **sub-class**
- **Constructor** in Java can be **overloaded**
- Overloaded methods must have a **different argument list**
- Overloaded method should always be the part of the same class (can also take place in sub class), with **same name** but **different parameters**
- They may have the **same** or **different return types**

Runtime Polymorphism

- Using of **override** method

```
public static void main(String[] args) {
  Rectangle rect = new Rectangle(3.0, 4.0);
  Rectangle square = new Square(4.0);

  System.out.println(rect.area());
  System.out.println(square.area());
}
```

Method overriding

Runtime Polymorphism (2)

SoftUni Foundation

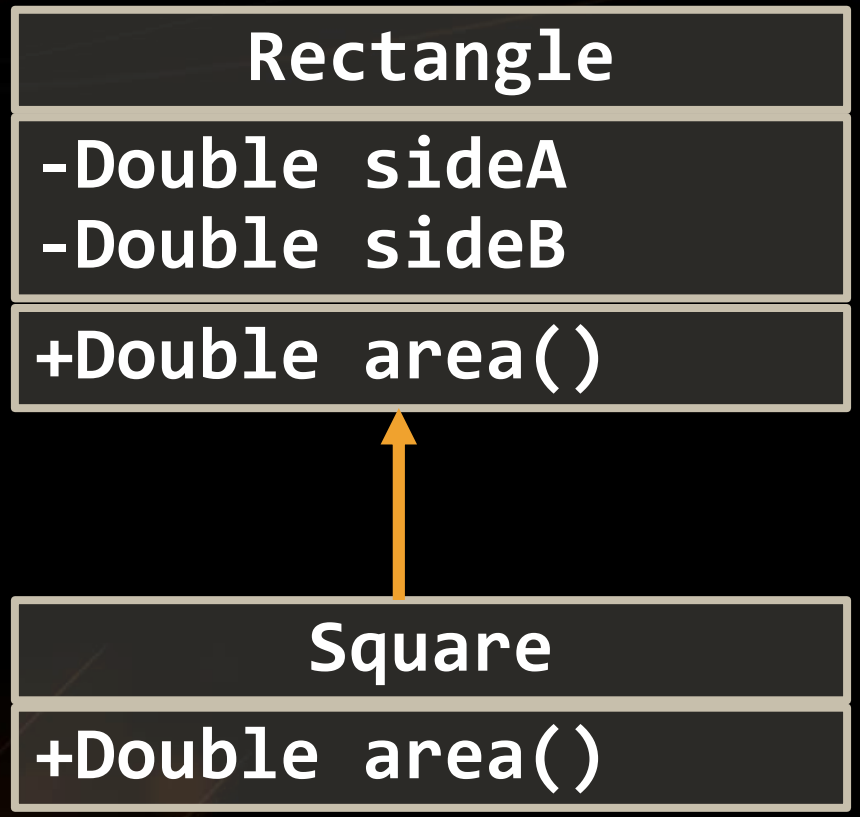
- Also known as **Dynamic Polymorphism**

```
public class Rectangle {
  public Double area() {
    return this.a * this.b;
  }
}
```

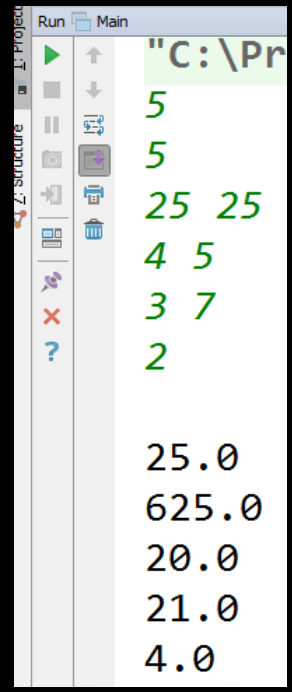
```
public class Square extend Rectangle {
  public Double area() {
    return this.a * this.a;
  }
}
```

Method overriding

Problem: Override Method




```
for (Rectangle rectangle : listOfRectangles) {
    System.out.println(rectangle.area());
}
```



Solution: Override Method

SoftUni Foundation

```
public class Square extends Rectangle {
   private Double sideA;
   public Square(Double side) {
      super(side);
      this.sideA = side * 2;
   }
   public Double perimeter() { return this.sideA * 4; }
   @Override
   public Double area() {
      return this.sideA * this.sideA;
   } }
```

Rules for Overriding Method

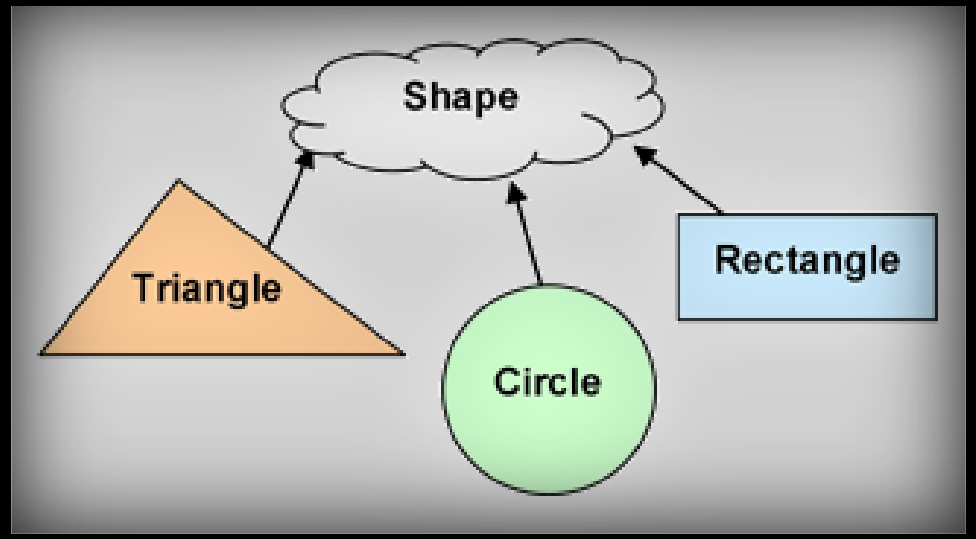
SoftUni Foundation

- **Overriding** can take place **sub-class**.
- **Argument list** must be the **same** as that of the **parent method**
- The overriding method must have **same return type**
- **Access modifier** cannot be more **restrictive**
- **Private, static and final** methods can **NOT** be overriden
- The overriding method **must not** throw new or broader **checked exceptions**

SoftUni Foundation

Polymorphism

Live Exercises in Class (Lab)



Abstract Classes

Abstract Classes

SoftUni Foundation

- Abstract class **can NOT be instantiated**

```
public abstract class Shape {}
public class Circle extends Shape {}
Shape shape = new Shape(); // Compile time error
Shape circle = new Circle(); // polymorphism
```

Abstract Classes (2)

SoftUni Foundation

- An **abstract** class **may or may not** include abstract **methods**
- If it has **at least one abstract method**, it must be declared **abstract**
- To use **abstract class**, you need to **extend it**

Abstract Classes Elements

SoftUni Foundation

```
Public abstract class Shape {
  private Point startPoint;
  protected Shape(Point startPoint) {
    this.startPoint = startPoint;
  }
  public getStartPoint() { return this.startPoint; }
  public abstract void draw();
}
```

Can have **fields**

Can have **constructors**

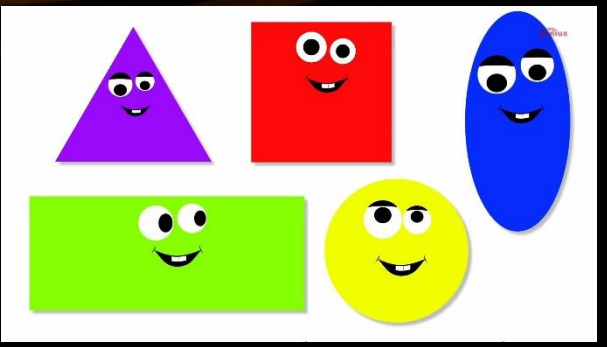
Can hold **methods** with code in them

Abstract methods **MUST** be **overridden**

Problem: Shapes

Encapsulate area

Shape
-Double perimeter -Double area
+getPerimeter() #setPerimeter(Double area) *+calculatePerimeter* *+calculateArea*



Rectangle
-Double height -Double width
+calculatePerimeter +calculateArea

Circle
-Double height -Double width
+calculatePerimeter +calculateArea

Solution: Shapes

SoftUni Foundation

```
public abstract class Shape {
  private Double perimeter;
  private Double area;
  protected void setPerimeter(Double perimeter) {
    this.perimeter = perimeter;
  }
  public Double getPerimeter() { return this.perimeter; }
  protected void setArea(Double area) {this.area = area;}
  public Double getArea() { return this.area; }
  protected abstract void calculatePerimeter();
  protected abstract void calculateArea();
}
```

Solution: Shapes

SoftUni Foundation

```
public class Rectangle extends Shape {
  //TODO: Add fields
  public Rectangle(Double height, Double width) {
    this.setHeight(height); this.setWidth(width);
    this.calculatePerimeter(); this.calculateArea(); }
  //TODO: Add getters and setters
  @Override
  protected void calculatePerimeter() {
    setPerimeter(this.height * 2 + this.width * 2); }
  @Override
  protected void calculateArea() {
  setArea(this.height * this.width); } }
```

Solution: Shapes

SoftUni Foundation

```
public class Circle extends Shape{
  private Double radius;
  public Circle (Double radius) {
    this.setRadius(radius);
    this.calculatePerimeter();
    this.calculateArea(); }
  public final Double getRadius() {
    return radius;
  }

  //TODO: Finish encapsulation
  //TODO: Override calculate Area and Perimeter
}
```

Summary

- What is **Polymorphism**?
- Types of Polymorphism
  - **Static** polymorphism
  - **Dynamic** polymorphism
- **Overload** Methods
- **Override** Methods
- Abstract Classes
- Abstract Methods

Polymorphism

SoftUni Foundation

Questions?

https://softuni.bg/courses/java-oop-basics

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Software University

SoftUni Foundation

FORUM